Amendments to the Specification

Please replace paragraphs [0061], [0062], [0063], [0064], [0065], [0068], [0072], [0083] and [0088] with the following amended paragraph:

[0061] The message broker 10 comprises a Web server with a multi-thread servlet engine 10a. The message broker 10 is provided with a first channel adapter 22, in this embodiment comprising a first channel adapter servlet 22a, a second channel adapter 23, in this embodiment comprising a second channel adapter servlet 23a and a plurality of channels 24 each addressable by the first channel adapter servlet 22a and second channel adapter servlet 23a. In the present example, the channel adapters 22, 23 are servlets which run within a thread allocated by the servlet engine 10a to process incoming HTTP requests. The servlets 22a, 23a are operable to run appropriate to perform a "push" or "pull" operation to place messages in the channel 24 and withdraw messages from the channel 24 respectively.

[0062] Referring now to Figures 1 and 2, the communication system works as follows. The subscriber module 18 is instructed to invoke the call back module 21 on occurrence of a new message. The subscriber module 18 sends a subscribe instruction 26 to the receiver module 19. The subscribe instruction 26 contain source information corresponding to a message channel to enable the message broker 21 to establish an appropriate communication link. The receiver module 19 generates a message request 27 in the form of an HTTP GET request, including the

source information from the subscribe instruction 26. The HTTP GET request is transmitted via the Internet link 14 to the message broker 10. In conventional manner the message broker servlet engine 10a receives the HTTP request and runs the second channel adapter servlet as specified in the HTTP request. The servlet 23a then processes the HTTP request, by reading the GET request 27 to obtain the source information to identify the relevant channel. In the present example, the source information is simply a channel identification name or number.

[0063] Once a channel has been identified, in this case the channel 28, the second channel adapter servlet 23a performs a pull operation 29 to attempt to pull an event from the channel 28. Where no event is found, i.e. no message has been placed in channel 28, no response will be sent by the second channel adapter servlet 23a until, in the present example, a pre-determined period of time has elapsed. In the present example, since the pull request is performed by a servlet 23a, if there is no information in the identified channel the thread running the servlet 23a will "sleep" until the standard time-out period elapses or notification of a message "push" is received. Once the predetermined time period has elapsed, the thread running the servlet 23a is woken up in conventional manner. The second channel adapter servlet 23a then sends a standard time-out response to the receiver module 19. In this example, the standard HTTP time-out error message will be sent, code 504, as the HTTP GET response 31. On receipt of the time-out response 31, the receiver module ~~20~~ 19 then promptly retransmits a GET

request 27′, and the second channel adapter servlet 23a on receipt of the HTTP GET request 27′ will once again attempt to pull a message from the channel 28. This cyclical process of the receiver module ~~20 transmission~~ 19 transmitting a GET request to the second channel adapter servlet 23a, receiving a time out response 31 and re-transmitting a GET request 27 may continue indefinitely until a message is received.

[0064] To send a message via the message broker 10, the publisher module 15 of the first client system 11 will generate message information, including destination information and content information i.e. the body of the message, and forward this message as a publish instruction 32 to the transmission module 16. The transmission module 16 will generate a message 33 in the form of an HTTP POST request and transmit this information via the firewall 17 and the Internet link 13 to the message broker 10. The message 33 is received by the message broker servlet engine 10a which runs the first channel adapter servlet 22a as specified in the HTTP request. The servlet 22a then processes the HTTP request by reading the destination information and identifies the appropriate channel in which the message should be placed. In the present example, as for the GET request source information, the destination information is simply a channel identification name or number. If no such channel exists, the first channel adapter servlet 22a may dynamically create the channel, i.e. allocate the channel name to one of the plurality of channels 24. In this example, the channel 28 is identified and the servlet 22a performs a push

operation 34 to place the message on the identified channel. The first channel adapter servlet 22 then sends a notification 35 is sent to any thread listening to that channel, in this example the second channel adapter servlet 23a.

[0065] Referring to Figure 2, the push operation 34 has now placed the message in the channel 28 within the predetermined time-out period from the pull operation 29′. The second channel adapter servlet 23a receives the notification 35 and acts to pull the message from the channel 28. The second channel adapter servlet 23a then transmits a standard response to the HTTP GET request 36, including at least the content information of the message 32. In this example, the receiver module ~~20~~ 19 then transmits a new event notification 37 to the call back module 21, including the message content information as part of the argument as part of the new event notification 37. In this example, the second receiver module ~~20~~ 19 then transmits a new HTTP GET request 27′′, the second channel adapter servlet 23a then transmits a further pull request 29′′ and the cycle continues.

[0068] In an alternative implementation, it might be envisaged that the functionality of the servlets 22a, 23a could be implemented instead at socket level. A thread processing a HTTP PUT request will check whether there is a socket connection associated with the message channel. If yes, the message is simply be sent to the client system using the socket connection information, and then the socket connection is removed. If there

is no socket connection, the message is stored in the message channel as discussed before. To retrieve a message, a thread processing an HTTP GET request checks the specified message channel. If a message is stored in the message channel, it is returned to the client system. Otherwise, the socket connection is stored as information associated with the message channel, and a time out specification placed in a time queue. If a message is pushed into the channel before time out occurs, the thread processing the HTTP PUT request simply sends the message to the client using the socket connection information as discussed above. In the event of a time out, the thread associated with the time out will wake up and retrieve the message channel identification name or number associated with the time out. If there is still socket connection information associated with the message channel, a time out response is sent to the client system using the socket connection and then the socket connection information is removed. If no socket connection information is associated with the message channel, this indicates that a message was sent to a client system in the interim. No action is then taken and the thread returns to the start of the process.

[0072] The first combined channel adapter ~~40~~ 38 comprises a first adapter element 22' operable to push messages onto channel 28' in the same manner as the first channel adapter servlet 22a, and a second adapter element 23' operable to pull messages from channel 28'' in the same manner as the second channel adapter

servlet 23a described above. Similarly, the second combined channel adapter ~~41~~ 39 comprises a first adapter element 22'' adapted to push messages onto channel 28'' in like manner to the first channel adapter servlet 22a described above, and a second adapter element 23'' operable to pull messages from channel 28' in the same manner of the second channel adapter servlet 23a described above. Using such an arrangement, each client system 11', 12' is operable both to transmit and receive messages via the message broker 10' using the method as described above in relation to Figure 1. Because the system of Figure 3 can be made transparent, it will be apparent that any suitable communication protocol may be used by the client systems 11', 12'.

[0083] The HTTP client adapter 49 has similar functionality to the combined adapter module 38, but with the additional functions of encapsulating the HTTP client request and providing authentication and authorisation. The browser 42 may be enabled to access the Intranet web server by provision of an appropriate cookie which provides the necessary information to the HTTP client adapter 49, for example as part of the cookie 56. The cookie could be electronically signed by the provider of the intranet web server 45 such that the user of the browser ~~42~~ 41a is happy to install the cookie 56 on his computer.

[0088] When is desired to provide the second client system 67 with remote access to the printer 65, the remote diagnostic support tool 66 is enabled from the PC 64 to address the printer

65 and to establish a link over the Internet connection 63. The message broker 60 establishes a bi-directional communication channel comprising message channels 60a, 60b, as described hereinbefore in relation to Figure 3. The PC 70 sends a device instruction to the remote control module 69. The remote control module generates a message encoded as an HTTP instruction comprising the device instruction, destination information identifying the message channel 60a and also device identification information as required. The remote control device 69 then sends the message through the firewall 68, and message broker 60 to the remote diagnostic support tool 66. The remote diagnostic support tool 66 transmits the device instruction to the printer 65. The printer 65 may return device information, in this example the printer make, serial number, and configuration. The RDST 66 then encodes the device information as an HTTP POST request, including destination information corresponding to the message channel 60b, and transmits this instruction to the remote control tool 69 via the message broker 60. This printer information is then transferred to the PC 70. It might be envisaged that the printer information be displayed as a simulated control panel, for example, ~~an~~ on display 71. The operator of the PC 70 is then able to transmit further appropriate device instructions, for example to reconfigure the printer or transmit appropriate software update via the remote control 69 and remote diagnostic support tool 66 to the printer 65. The updated printer configuration can then be retrieved by the remote diagnostic support tool 66 and

retransmitted to the remote control 69 as before. It might also be envisaged that, for example, updated driver software might be sent to the PC 64. Once the session has ended, the connections 63a, 63b to the message broker 60 are dropped. Such an arrangement permits nearly real time remote control of the printer 65 via the Internet.